T.Y. Diploma : Sem.VI
[CO/CD/CM]

Software Testing

Time : 3 Hrs.] Prelim Question Paper Solution **[Marks: 100**

Q.1(a) Attempt any THREE of the following : [12]

Q.1(a) (i) Distinguish between Alpha testing and Beta testing. [4]

(A) Difference between Alpha Testing & Beta Testing

	Alpha Testing	Beta Testing
Place	Alpha testing is conducted within the organization.	Beta Testing is conducted at the client's place i.e. outside the organization.
Presence of Developer	In Alpha testing, developer is present there while testing.	In Beta testing, developer is not present while testing as it is tested at client's side
When to Conduct?	When the development of software is near to completion stage, Alpha testing is conducted.	After passing Alpha testing and before the final launching or releasing the software, Beta testing is conducted.
Recording of Errors	In Alpha testing, developer is present while testing so he records all the problems and errors encountered during testing.	In Beta Testing, customer records all the errors and other issues encountered during this testing and reports to the developer.
Who will test? (Who will test?)	Alpha testing is conducted within the organization and tested by representative group of end users at the developer's side and sometimes by Independent team of testing.	Beta testing is conducted by the end users or other persons and they are not programmers, software engineers or testers.
Environment	This testing is conducted in virtual & controlled environment.	This testing is conducted in real time or live environment.
Public Involvement	This testing is close for public.	This testing is open for public

Q.1(a) (ii) Differentiate between quality assurance and quality control. [4]

(A)

	Quality Assurance	Quality control
1)	Concentrates on the process of producing the products	Concentrates on specific products
2)	Defect-prevention oriented	Defect-detection and correction oriented
3)	Usually done throughout the life cycle	Usually done after the product is built
4)	This is usually a staff function	This is usually a line function
5)	Examples – reviews and audits	Examples – software testing at various levels

Q.1(a) (iii) Explain the terms error and bugs and failure? [4]

(A)

- If someone makes an error or mistake in using the software, this may lead directly to a problem.
- People also design and build the software and they can make mistakes during the designing and building phase.
- These mistakes means that there are flaws in the software itself. These are called as defects or bugs or faults.
- **Error** – A human action that produces an incorrect result.
- **Defect (bug, fault)** - A flaw in a component or system that can cause the component or system to fail to perform its required function.

A defect is an error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results. A defect is said to be detected when a failure is observed.

- **Failure** – Deviation of the component or system from its expected result or service.
  Failure is the inability of a system or a component to perform its required functions within specified performance requirements. Failure occurs when fault executes.

Defects and failures arise from –

- Errors in the specification, design and implementation of the software.
- Errors in use of the system
- Environmental conditions
- Intentional damage
- Potential consequences of earlier errors, intentional damage, defects or failures.

Q.1(a) (iv) List and explain any four testing skills for software tester. [4]

(A)

- **Concept of Testing :** A tester must have complete knowledge about testing as a discipline. He/she must understand methods, processes and concepts of testing. He\/she must be capable of doing test planning, designing test scenario, writing test cases and defining test strategy, and defining test data.

- **Levels of testing :** Testing is a multitier activity where the application goes from one level to another after successful completion of the previous level. Testers are involved in each phase of software development right from proposal and contract, followed by requirement till acceptance testing. Testers must ensure that each phase is passes successfully.

- **Techniques for Validation and Verification :** Techniques of verification/validation must be understood and facilitated by testers to the development team, customer and management. While writing test cases, he/she needs to define test case pass/fail criteria to validate the test case and product.

- **Selection and use of Testing Tools :** Testing involves use of various tools including automation tools, defect tracking tools, configuration management tools and simulators. A tester must understand and use the tools effectively.

- Knowledge of Testing standards
- Risk assessment and management
- Developing test plan
- Defining acceptance criteria
- Checking of testing processes
- Execution of test plan
- Continuous improvement of testing process

Q.1(b) Attempt any ONE of the following : [6]

Q.1(b) (i) Describe any two types of Defect in detail. [6]

(A) Defects are classified in the following ways :

1) **Requirement Defects :** Requirement related defects arise in a product when one fails to understand what is required by the customer. These defects may be due to customer gap, where customer is unable to define his requirements, or producer gap, where developing team is not able to make a product as per requirements.

   These defects are further classified as follows –

   - **Functional defects -** these defects are mainly about the functionalities present/absent in the application which are expected/not expected by the customer. Generally, system testing concentrates on functionality as the first part of testing. Non-working functions and functions not provided as required may represent functional defects.

- **Interface Defects** – these defects talk about various essential as per the requirement statement. Generally, these defects may be due to user interface problems, problems with connectivity to other systems including hardware, tec.

2) **Design Defects :** Design defects generally refer to the way of design creation or its usage while creating a product. The customer may or may not be in a position to understand these defects, if structures are not correct. They may be due to problems with design creation and implementation during software development life cycle.

They may be classified as follows :

- **Algorithmic defects** – these defects may be introduced in a product if designs of various decisions are not handled correctly. Some applications have a control over complexity and usage of various algorithms to translate requirements correctly into coding. Various permutations and combinations may cause defect introduction in the product.
- **Module interface defects** – these are about communication problem between various modules. If one module gives some parameters which are not recognized by another, it creates module-interface defects.
- **System-interface defects** – these defects may be generated when application communication with environmental factors is hampered. System may not be able to recognize inputs coming from the environment, or may not be able to give outputs which can be used by the environment.
- User-Interface defects – these defects may be a part of system-interface defects where the other system working with the application is a human being. User-interface defects may be a part of navigation, look and feel of defects which affect usability of an application.

(Coding Defects – Variable Declaration/Initialization defect
Database related defects
Commenting/documentation defects

Testing defects – test-design defects
Test environment defects
Test tool defects)

Q.1(b) (ii) Explain Inspection Process. [6]

(A) Inspection

- The document under inspection is prepared and checked thoroughly by the reviewers before the meeting comparing the work product with its sources and other referenced documents and using rules and checklist.
- In the inspection meeting, the defects found are logged and any discussion is postponed until the discussion phase.
- The goals of inspection are –
  - Help the author to improve the quality of the document under inspection.
  - Remove defects immediately.
  - Improve product quality, by producing documents with a higher level of quality.
  - Create a common understanding by exchanging information among the inspection participants.
  - Train new employees in organization's development process.
  - Learn from defects found and improve processes in order to prevent recurrence of similar defects.
- Key characteristics of an inspection are :
  - It is led by a trained moderator and not by the author.
  - It involves peers to examine the product.
  - Rules and checklists are used during the preparation phase.

- A separate preparation is carried out during which the product is examined and the defects are found.
- The defects found are documented in a logging list.
- A formal follow-up is carried out by the moderator applying exit criteria.

Q.2 Attempt any FOUR of the following : [16]

Q.2(a) Difference between White box and Black box testing. [4]

(A)

Criteria	Black Box Testing	White Box Testing
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester.	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
Levels Applicable To	Mainly applicable to higher levels of testing : Acceptance Testing System Testing	Mainly applicable to lower levels of testing : Unit Testing Integration Testing
Responsibility	Generally, independent Software Testers	Generally, Software Developers
Programming Knowledge	Not Required	Required
Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design

Q.2(b) What is Test Plan? Write any two advantages of test planning. [4]

(A) A test plan is a document describing the scope, approach, objectives, resources, and schedule of a software testing effort. It identifies the items to be tested, items not be tested, who will do the testing, the test approach followed, what will be the pass/fail criteria, training needs for team, the testing schedule etc.

Advantages of test planning :

- Serves as a guide to testing throughout the development.
- We only need to define test points during the testing phase.
- Serves as a valuable record of what testing was done.
- The entire test plan can be reused if regression testing is done later on.
- The test plan itself could have defects just like software!

Q.2(c) Explain Security Testing. [4]

(A) Security Testing

- It is a type of non-functional testing.
- Security testing is basically a type of software testing that's done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
- It is a process to determine that an information system protects data and maintains functionality as intended.
- The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software's and hardware's and firewall etc.
- Software security is about making software behave in the presence of a malicious attack.
- The six basic security concepts that need to be covered by security testing are: confidentiality, integrity, authentication, availability, authorization and non-repudiation.

Security Testing - Techniques:

- Injection
- Broken Authentication and Session Management
- Cross-Site Scripting (XSS)
- Insecure Direct Object References
- Security Misconfiguration
- Sensitive Data Exposure
- Missing Function Level Access Control
- Cross-Site Request Forgery (CSRF)
- Using Components with Known Vulnerabilities
- Unvalidated Redirects and Forwards

Q.2(d) What are the different points to be noted in reporting defects? [4]

(A) It is essential that you report defects effectively so that time and effort is not unnecessarily wasted in trying to understand and reproduce the defect. Here are some guidelines:

(i) Be specific

- Specify the exact action: Do not say something like 'Select Button B'.
- Do you mean 'Click Button B' or 'Press ALT+B' or 'Focus on Button B and click ENTER'.
- In case of multiple paths, mention the exact path you followed: Do not say something like "If you do 'A and X' or 'B and Y' or 'C and Z', you get D." Understanding all the paths at once will be difficult. Instead, say "Do 'A and X' and you get D." You can, of course, mention elsewhere in the report that "D can also be got if you do 'B and Y' or 'C and Z'."
- Do not use vague pronouns: Do not say something like "In Application A, open X, Y, and Z, and then close it." What does the 'it' stand for? 'Z' or, 'Y', or 'X' or 'Application A'?"

(ii) Be detailed

- Provide more information (not less). In other words, do not be lazy.
- Developers may or may not use all the information you provide but they sure do not want to beg you for any information you have missed.

(iii) Be objective

- Do not make subjective statements like "This is a lousy application" or "You fixed it real bad."
- Stick to the facts and avoid the emotions.

(iv) Reproduce the defect

- Do not be impatient and file a defect report as soon as you uncover a defect. Replicate it at least once more to be sure.

(v) Review the report

- Do not hit 'Submit' as soon as you write the report.
- Review it at least once.
- Remove any typing errors.

Q.2(e) Describe the Statement Coverage in White Box Testing. [4]

(A) Statement Coverage

The first kind of logic coverage can be identified in the form of statements. It is assumed that if all the statements of the module are executed once, every bug will be noticed.

Example :

```
scanf("%d", &x);
scanf("%d", &y);
while (x != y)
{
    if(x > y)
        x = y - y;
    else
        y = y - x;
}
```

```
printf("x = " , x);
printf("y = " , y);
```

if every statement is to be covered, then the following test cases must be designed.
Test case 1: x=y=n, where n is any number
Test case 2: x=n=n` , where n` and n are different numbers
Test case 1 just skips the while loop and all loop statements are not executed. Considering test case 2, the loop is also executed. However, every statement inside the loop is not executed. So two more cases are designed.
Test case 3 : x>y
Test case 4: x<y

These test cases will cover every statement in the code segment, however statement coverage is a poor criteria for logic coverage. Test case 3 and 4 are sufficient to execute all the statements in the code. Bit, if only test cases3 and 4 are executed, then conditions and paths in test cse 1 will never be tested and errors will go undetected. Thus, statement coverage is necessary but not sufficient criteria for logic coverage.

Q.3 Attempt any FOUR of the following : [16]

Q.3(a) Illustrate process of Decision Table with an example. [4]

(A) Decision tables

Decision tables are a good way to describe requirements when there are several business rules that interact together. Using decision tables it becomes easier for the requirements specialist to write requirements which cover all conditions. As to the tester, it becomes easier for them to write complete test cases.

Example : A customer requests a cash withdrawal. One of the business rules for the ATM is that the ATM machine pays out the amount if the customer has sufficient funds in their account or if the customer has the credit granted. Already, this simple example of a business rule is quite complicated to describe in text. A decision table makes the same requirements clearer to understand:

Conditions	**R1**	**R2**	**R3**
Withdrawal Amount <= Balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

Write test cases

Write test cases based on the table. At least one test case per column gives full coverage of all business rules.

- Test case for R1: balance = 200, requested withdrawal = 200. Expected result: withdrawal granted.
- Test case for R2: balance = 100, requested withdrawal = 200, credit granted. Expected result: withdrawal granted.
- Test case for R3: balance = 100, requested withdrawal = 200, no credit. Expected Result: withdrawal denied.

Q.3(b) State process of Bi-Directional/Sandwich Integration testing with labeled diagram. [4] Give its any two advantages and disadvantages.

(A) Sandwich Testing

Sandwich testing defines testing into two parts, and follows both parts starting from both ends i.e., top-down approach and bottom-up approach either simultaneously or one after another. It combines the advantages of bottom-up testing and top-down testing at a time. Figure shows a sandwich testing approach.

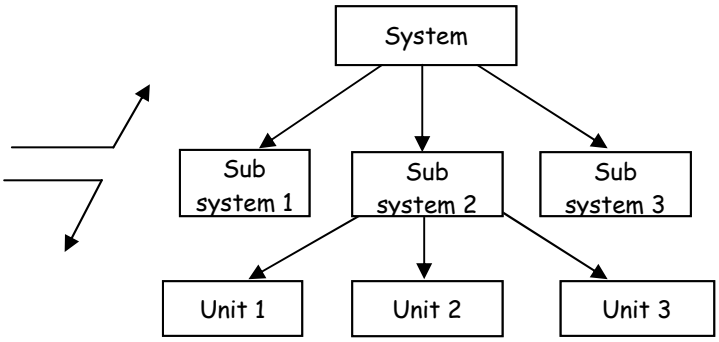

Fig. : Sandwich testing approach

Process of Sandwich Testing

- Bottom-up testing starts from middle layer and goes upward to the top layer. Generally for very big systems, bottom-up approach starts at subsystem level and goes upwards.
- Top-down testing starts from middle layer and goes downward. Generally for vary big systems, top-down approach starts at subsystem level and goes downwards.
- Big-bang approach is followed for the middle layer. From this layer, bottom-up approach go and top-down approach goes downwards.

Advantages of sandwich testing

- Sandwich approach is useful for very large projects having several subprojects. When development follows a spiral model and the module itself is as large as a system, then one can use sandwich testing.
- Both top-down and bottom-up approaches start at a time as per development schedule. Units are tested and brought together to make a system. Integration is done downwards.
- It needs more resources and big teams for performing both methods of testing at a time or one after the other.

Disadvantages of sandwich testing

- It represents very high cost of testing as lot of testing is done. One part has top-down approach while another part has bottom-up approach.
- It cannot be used for smaller systems with huge interdependence between different modules. It makes sense when the individual subsystem is as good as complete system.
- Different skill sets are required for testers at different levels as modules are separate system handling separate domains like ERP products with modules representing different functional areas.

Q.3(c) Explain regression Testing. [4]

(A) Regression testing

- The regression test is a retest of a previously tested program following modification, to ensure that faults have not been introduced or uncovered as a result of the changes made.
- Such faults often arise as unplanned side effects of program changes. Regression testing may be performed at all levels and applies to functional, non-functional and structural testing.
- The purpose of regression testing is to verify that modifications in the software or the environment have not caused unintended adverse effects.
- Test cases that are used in regression testing run many times and thus have to be well documented and reusable.
- It must be determined how extensive a regression test has to be. There are the following possibilities :
  - Rerunning of all the tests that have detected faults, which have been fixed in the new software release.(defect retest, confirmation testing)

- Testing of all program parts that were changed or corrected (testing of altered functionality)
- Testing of all program parts or elements that were newly integrated. (testing of new functionality)
- Testing of the whole system (complete regression test)

- Selection of regression test cases –
  - Only the high priority test according to the test plan are repeated.
  - In the functional test, certain variations are omitted.
  - Restrictions of the tests to certain configurations only(e.g. testing of the English product version only, testing of one operating system version only).
  - Restriction of the tests to certain subsystems or test levels.

Q.3(d) Define software metrics. Describe Product Vs. Process metrics and Objective. [4]

(A) Software metrics can be defines as 'the continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products.

Describe Product Vs. Process metrics

Software metrics may be broadly classified as either product metrics or process metrics. Product metrics are measures of the software product at any stage of its development, from requirements to installed system. Product metrics may measure the complexity of the software design, the sixe of the final program, or the number of pages of documentation produced.

Process metrics, on the other hand, are measures of the software development process, such as the overall development time, type of methodology used, or the average level of experience of the programming staff.

Objective Vs. Subjective Metrics

Objective measures should always result in identical value for a given metric, as measured by two or more qualified observers. For subjective measures, even qualified observers may measure different values for a given metric. For example, for product metric, the size of the product measured in line of code (LOC) is an objective measure. In process metrics, the development time is an example of objective measure, while the level of a programmer's experience is likely to be a subjective measure.

Q.3(e) Explain activities performed by Static and Dynamic testing tools. [4]

(A) Activities performed by Static testing tools :

And Static tools parse the program text, recognize the various sentences and detect the following:
Statements are well-formed
Inferences about the control flow of the program
Compute the set of all possible values for program data.

Control flow analysis – this analysis detects loops with multiple exists and entry points and unreachable code.
Data use analysis – it detects all types of data faults.
Interface analysis – it detects all interface faults. It also detects functions which are never declared and never called or function results that are never used.
Path analysis – it identifies all possible paths through the program and unravels the program's control.

Activities performed by Dynamic testing tools –

- List the number of times a component is called or line of code is executed. This information about the statement or path coverage of their test cases is used by testers.

- Report on whether a decision point has branched in all directions, thereby providing information about branch coverage.
- Report summary statistics providing a high-level view of the percentage of statements, paths and branches that have been covered by the collective set of test cases run. This information is important when test objectives are stated in terms of coverage.

Q.4(a) Attempt any THREE of the following : [12]

Q.4(a) (i) Write the benefits of Automation. [4]

(A) **Reduction of testing effort** – In verification and validation strategies, numerous test case design methods have been studied. Test cases for a complete software may be hundreds of thousands or more in number. Executing all of them manually takes a lot of testing effort and time. Thus, execution of test suits through software tools greatly reduces the amount of time required.

Reduces the testers' involvement in executing tests – Something executing the test cases takes a long time. Automating this process of executing the test suit will relieve the testers to do some work, thereby increasing the parallelism in testing efforts.

Facilitates regression testing – regression testing is the most time consuming process. If we automate the process testing, then testing effort as well as time taken will reduce as compared to manual testing.

Avoids human mistakes – manually executing the test cases may incorporate errors in the process or sometimes, we may be biased towards limited test cases while checking the software. Testing tools will not cause these problems which are introduced due to manual testing.

(or write any from the following –

- Reduced overall cost of the software
- Simulated testing
- Internal testing
- Test enablers
- Test case design)

Q.4(a) (ii) Write a note on costs incurred in Testing Tools. [4]

(A) **Automated script development** – automated test tools do not create test scripts. Therefore, a significant time is needed to program the tests. Scripts are themselves programming languages. Thus, automating test execution requires programming exercises.

Training is required - it is not clear that the tester will be aware of all the tools and can use them directly. He may require training regarding the tool, otherwise it ends up on the shelf or implemented inefficiently. Therefore, it becomes necessary that in a new project, cost of training on the tools should also be in the project budget and schedule.

Configuration management – configuration management is necessary to track large number of files and test related artifacts.

Learning curve for the tools – there is a learning curve in using any new tool. For ex. test scripts generated by the tool during recording must be modified manually, requiring tool-scripting in order to make the script robust, reusable and maintainable.

(following points can also be added – but any 4 are required
Testing tools can be intrusive
Multiple tools are required)

Q.4(a) (iii) State the factors that need to be considered to identify resource requirement by test manager? List five test deliverables for test plan. **[4]**

(A) Following factors need to be considered –

- Machine configuration (RAM, processor, disk) needed to run the product under test.
- Overheads required by the test automation tool, if any
- Supporting tools, such as compilers, test data generators, configuration management tools and so on.
- The different configurations of the supporting software that must be present.
- Special requirements for running machine-intensive tests such as load tests and performance tests.
- Appropriate number of licenses of the software.

The different types of Test deliverables are:

1) The test plane itself (master test plan, and various other test plans for the project)
2) Test case design specifications.
3) Test cases, including any automation that is specified in the plan.
4) Test logs produced by running the tests.
5) Test summary reports.

Q.4(a) (iv) What are the different techniques for finding defects? **[4]**

(A) Static techniques

Static techniques of quality control define checking the software product and related artifacts without executing them. It is also termed as 'desk checking/ verification/ white box testing'. It may include reviews, walkthroughs, inspection, and audits. Here, the work product is reviewed by the reviewer with the help of a checklist, standards and other artifacts, knowledge and experience, in order to locate the defect with respect to the established criteria. Static technique is so named because it involves no execution of code, product, documentation etc. This technique helps in establishing 'conformance to requirements' view.

Dynamic testing

Dynamic testing is a validation technique which includes dummy or actual execution of work products to evaluate it with expected behavior. It includes black box testing methodology such as system testing and unit testing. The testing methods evaluate the product with respect to requirements defined, designs created and mark it as 'pass' or 'fail'. This technique establishes 'fitness for use' view.

Operational technique

Operational techniques typically include auditing work products and projects to understand whether the processes defined for development/testing are being followed correctly or not, and also whether they are effective or not. It also includes revisiting the defects before and after fixing and analysis.

Q.4(b) Attempt any ONE of the following : **[6]**

Q.4(b) (i) Explain Defect management process. **[6]**

(A) Defect Management Process (Approach)

Defects found during verification and validation process in software development life cycle must be recorded so that it helps in further analysis and root causes of the defect. The defect are used for analysing development process as well as test process statistically, so that corrective and preventive actions can be initiated to make the processes strong.

Defect management process must have the following characteristics.

- Primary Goal of Defect Management is to Prevent Defect.
- Defect Management Must Be Risk Driven.
- Defect Prevention Must Be an Integral Part of Development Process.
- Process of Defect Tracking Must Be Automated As Maximum As Possible.
- Defect Information Must Be used for Process Improvement.

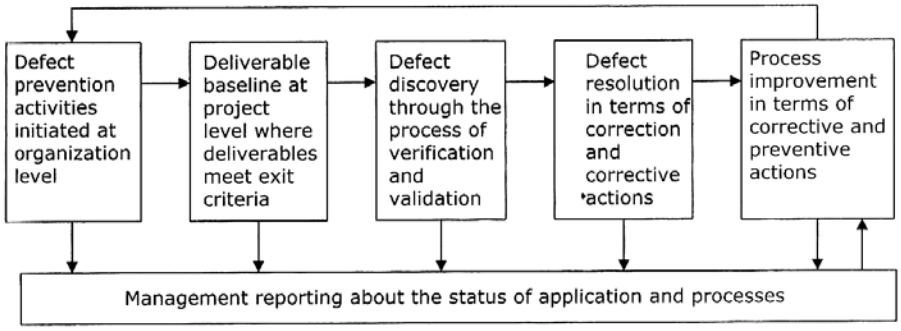

Fig.1: Defect Management Process

Q.4(b) (ii) State two objectives of user documentation testing? Mention any benefits of user documentation testing. **[6]**

(A) User Documentation testing

User documentation covers all the manuals, user guides, installation guides, setup guides, read me file, software release notes, and online help that are provided along with the software to help the end user to understand the software system.

User documentation testing should have two objectives.

(i) To check if what is stated in the document is available in the product.

(ii) To check if what is there in the product is explained correctly in the document.

User documentation testing focuses on ensuring what is in the document exactly matches the product behavior, by sitting in front of system and verifying screen by screen, transaction by transaction and report by report. In addition, user documentation testing also checks for the language aspects of the document like spell check and grammar.

Some of the benefits of user documentation testing are:

(i) User documentation testing aids in highlighting problems over looked during reviews.

(ii) High quality user documentation ensures consistency of documentation and product, thus minimizing possible defects reported by customers. It also reduces the time taken for each support call – sometimes the best way to handle a call is to alert the customer to the relevant section of the manual. Thus the overall support cost is minimized.

(iii) Results in less difficult support calls. When a customer faithfully follows the instructions given in a document but is unable to achieve the desire (or promised) results, it is frustrating and often this frustration shows up on the support staff. Ensuring that a produce is tested to work as per the document and that it works correctly contributes to better customer satisfaction and better morale of support staff.

(iv) New programmers and testers who join a project group can use the documentation to learn the external functionality of the product.

(v) Customers need less training and can proceed more quickly to advanced training and product usage if the documentation is of high quality and is consistent with the product. The high – quality documentation can result to a reduction of overall training costs for user organizations.

Q.5 Attempt any TWO of the following : **[16]**

Q.5(a) Explain criteria's for selecting test tools. **[8]**

(A) Criteria for Selecting Test Tools

The Categories for selecting Test Tools are,

(1) Meeting requirements;

(2) Technology expectations;

(3) Training/skills;

(4) Management aspects.

(1) Meeting requirements

There are plenty of tools available in the market but rarely do they meet all the requirements of a given product or a given organization. Evaluating different tools for different requirements involve significant effort, money, and time. Given of the plethora of choice available, huge delay is involved in selecting and implementing test tools.

(2) Technology expectations

Test tools in general may not allow test developers to extends/modify the functionality of the framework. So extending the functionality requires going back to the tool vendor and involves additional cost and effort. A good number of test tools require their libraries to be linked with product binaries.

(3) Training/skills

While test tools require plenty of training, very few vendors provide the training to the required level. Organization level training is needed to deploy the test tools, as the user of the test suite are not only the test team but also the development team and other areas like configuration management.

(4) Management aspects

A test tool increases the system requirement and requires the hardware and software to be upgraded. This increases the cost of the already- expensive test tool.

OR

Guidelines for selecting a tool:

(1) The tool must match its intended use. Wrong selection of a tool can lead to problems like lower efficiency and effectiveness of testing may be lost.
(2) Different phases of a life cycle have different quality-factor requirements. Tools required at each stage may differ significantly.
(3) Matching a tool with the skills of testers is also essential. If the testers do not have proper training and skill then they may not be able to work effectively.
(4) Select affordable tools. Cost and benefits of various tools must be compared before making final decision.
(5) Backdoor entry of tools must be prevented. Unauthorized entry results into failure of tool and creates a negative environment for new tool introduction.

Q.5(b) With the help of example Boundary Value Analysis. [8]

(A) Most of the defects in software products hover around conditions and boundaries. By conditions, we mean situations wherein, based on the values of various variables, certain actions would have to be taken. By boundaries, we mean "limits" of values of the various variables.

- This is one of the software testing technique in which the test cases are designed to include values at the boundary. If the input data is used within the boundary value limits, then it is said to be Positive Testing. If the input data is picked outside the boundary value limits, then it is said to be Negative Testing.
- Boundary value analysis is another black box test design technique and it is used to find the errors at boundaries of input domain rather than finding those errors in the center of input.
- Each boundary has a valid boundary value and an invalid boundary value. Test cases are designed based on the both valid and invalid boundary values. Typically, we choose one test case from each boundary.
- Same examples of Boundary value analysis concept are:
  One test case for exact boundary values of input domains each means 1 and 100. One test case for just below boundary value of input domains each means 0 and 99. One test case for just above boundary values of input domains each means 2 and 101.
- **For Example:** A system can accept the numbers from 1 to 10 numeric values. All other numbers are invalid values. Under this technique, boundary values 0, 1, 2, 9, 10, 11 can be tested.
- Another Example is in exam has a pass boundary at 40 percent, merit at 75 percent and distinction at 85 percent. The Valid Boundary values for this scenario will be as follows:
  49, 50 - for pass

74, 75 - for merit
84, 85 - for distinction
Boundary values are validated against both the valid boundaries and invalid boundaries. The Invalid Boundary Cases for the above example can be given as follows
0 - for lower limit boundary value
101 - for upper limit boundary value

- Boundary value analysis is a black box testing and is also applies to white box testing. Internal data structures like arrays, stacks and queues need to be checked for boundary or limit conditions; when there are linked lists used as internal structures, the behavior of the list at the beginning and end have to be tested thoroughly.
- Boundary value analysis help identify the test cases that are most likely to uncover defects.

Q.5(c) Explain following concept related to web-based testing : [8]

(i) Usability testing (ii) Compatibility testing

(A) Usability testing related to web-based testing:

The presentation design emphasizing the interface between user and web application gives rise to usability testing.

The general guideline for usability testing

(1) Do not force users to remember key information across document
(2) Present information in a natural and logical order
(3) All possible options like menus, links or buttons on web pages should be visible and accessible from all web pages.
(4) Check that link are active such that there are no erroneous or misleading links
(5) Indicate similar concept through identical terminal and graphics.
(6) Provide understandable instruction where useful.
(7) The user should not get irritated while navigating through the web application. eliminate information which is irrelevant or distracting.
(8) Content writer should not mix the topics of information. There should be clarity in the information being displayed
(9) Organize information hierarchically, with more general information appearing before more specific details encourage the user to delve as deeply as needed, but to stop whenever sufficient information has been received.

Compatibility testing related to web-based testing:

Web application may be put in different environment when the user is using them in production. Server may be in different hardware, software or operating system environment than the recommended one. Client browsers may differ significantly from the expected environmental variable. Testing must ensure that performance is maintained on the range of hardware and software configuration and user must be adequately protected in case of configuration mismatch.

The general guideline for compatibility testing:

(1) There are number of different browsers and browser options. The web application has to be designed to be compatible for majority of the browser.
(2) The graphics have and other objects on website have to be tested on multiple browser.
(3) There are different versions of HTML. Also there are other code has to be tested like Java, Javascript, ActiveX etc.
(4) Test your web application on different operating systems like windows, Unix, MAC, Linux.

Q.6 Attempt any FOUR of the following : [16]

Q.6(a) Explain entry and exit criteria. [4]

(A) Entry Criterion

Entry criterion is used to determine when a given test activity should start. It also includes the beginning of a level of testing, when test design or when test execution is ready to start.

Examples:

- Verify if the Test environment is available and ready for use.
- Verify if test tools installed in the environment are ready for use.
- Verify if Testable code is available.
- Verify if Test Data is available and validated for correctness of Data.

Exit Criterion

Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.

Exit criterion should be part of test plan and decided in the planning stage.

Examples:

- Verify if All tests planned have been run.
- Verify if the level of requirement coverage has been met.
- Verify if there are NO Critical or high severity defects that are left outstanding.
- Verify if all high risk areas are completely tested.
- Verify if software development activities are completed within the projected cost.
- Verify if software development activities are completed within the projected timelines.

Q.6(b) State how to minimize risk impact while estimating defect? [4]

(A) Risk is a product of probability, impact and detention ability. Risk minimization has three different methods of handling its probability, impact or detection ability. Minimization of problem due to risk happens in the following manner.

- **Eliminate Risk**
  Elimination of risk involves taking steps to remove risk from the root. Risk's probability is reduced to almost '0' by removing the causes of risk. Preventive controls can eliminate the probability of risk to a large extent.
- **Mitigation of Risk**
  Actions initiated by an organization to minimize the possible damage due to realization of risk are considered 'mitigation actions'. Mitigation actions are planned by an organization so that if the risk is realized, the impact due to it can be reduced to minimum possible. The corrective controls used form the mitigation action. Corrective controls may be auto-corrective or suggestive.
- **Detection ability Improvements**
  Impact of a risk is more, if it catches the user unprepared. If people are aware of the risks, they can be well prepared to handle them. Generally, detective controls are used to increase the visibility towards risks. sometimes, detective controls give threshold to corrective controls.
- **Contingency Planning**
  Contingency planning refers to the actions initiated by an organization, when preventive or corrective actions fail and risk actually occurs. They are previously planned ways of tacking risks when all other planned activities for reducing probability and impact of the risk fail, and the risk becomes reality.

Q.6(c) What are the types of test reports? [4]

(A) Test reporting is a means of achieving communication through the testing cycle. There are 3 types of test reporting.

(i) Test incident report:

A test incident report is communication that happens through the testing cycle as and when Defects are encountered .A test incident report is an entry made in the defect repository each defect has a unique id to identify incident .The high impact test incident are Highlighted in the test summary report.

(ii) Test cycle report:

A test cycle entails planning and running certain test in cycle, each cycle using a different build of the product .As the product progresses through the various cycles it is expected to stabilize.

Test cycle report gives

(1) A summary of the activities carried out during that cycle.
(2) Defects that are uncovered during that cycle based on severity and impact.
(3) Progress from the previous cycle to the current cycle in terms of defect fixed.
(4) Outstanding defects that not yet to be fixed in cycle.
(5) Any variation observed in effort or schedule.

(iii) Test summary report:

The final step in a test cycle is to recommend the suitability of a product for release. A report that summarizes the result of a test cycle is the test summary report.

There are two types of test summary report:

(1) Phase wise test summary, which is produced at the end of every phase.
(2) Final test summary report.

A Summary report should present :

(1) Test Summary report Identifier.
(2) Description : Identify the test items being reported in this report with test id.
(3) Variances: Mention any deviation from test plans, test procedures, if any.
(4) Summary of results: All the results are mentioned here with the resolved incidents and their solutions.
(5) Comprehensive assessment and recommendation for release should include Fit for release assessment and recommendation of release.

Q.6(d) Illustrate process of Equivalence Partitioning with example. [4]

(A) Equivalence partitioning

- Equivalence Partitioning also called as equivalence class partitioning. It is abbreviated as ECP. It is a software testing technique that divides the input test data of the application under test into each partition.
- An advantage of this approach is it reduces the time required for performing testing of a software due to less number of test cases.
- In short it is the process of taking all possible test cases and placing them into classes. One test value is picked from each class while testing.
- Example : If you are testing for an input box accepting numbers from 1 to 1000 then there is no use in writing thousand test cases for all 1000 valid input numbers plus other test cases for invalid data.
- Using equivalence partitioning method above test cases can be divided into three sets of input data called as classes. Each test case is a representative of respective class.
- So in above example we can divide our test cases into three equivalence classes of some valid and invalid inputs.
- Test cases for input box accepting numbers between 1 and 1000 using Equivalence Partitioning:
  1) One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If you select other values between 1 and 1000 then result is going to be same. So one test case for valid input data should be sufficient.
  2) Input data class with all values below lower limit. i.e. any value below 1, as a invalid input data test case.
  3) Input data with any value greater than 1000 to represent third invalid input class.
- So using equivalence partitioning you have categorized all possible test cases into three classes. Test cases with other values from any class should give you the same result.
- We have selected one representative from every input class to design our test cases. Test case values are selected in such a way that largest number of attributes of equivalence class can be exercised.
- Equivalence partitioning uses fewest test cases to cover maximum requirements.

Q.6(e) State contents of test plan template. **[4]**

(A) (1) Introduction

(i) Scope

What features are to be tested and what features will not be tested what combinations of environment are to be tested and what not.

(2) References

(3) Test Methodology and Strategy/Approach

(4) Test Criteria

(i) Entry Criteria
(ii) Exit Criteria
(iii) Suspension Criteria
(iv) Resumption Criteria

(5) Assumptions, Dependencies, and Risks

(i) Assumption
(ii) Dependencies
(iii) Risk and Risk Management Plans

(6) Estimations

(i) Size Estimate
(ii) Effort Estimate
(iii) Schedule Estimate

(7) Test Deliverables and Milestones

(8) Responsibilities

(9) Resource Requirement

(i) Hardware Resources
(ii) Software Resources
(iii) People Resources
(iv) Other Resources

(10) Training Requirements

(i) Detail of Training Required
(ii) Possible Attendees
(iii) Any Constrains

(11) Defect Logging and Tracking Process

(12) Metrics Plan

(13) Product Release Criteria

❑ ❑ ❑ ❑ ❑